**Aim:** Write a program to build ConvNet using Transfer Learning approach

**Description:**

Transfer Learning leverages pre-trained models on large datasets (like ImageNet) to solve similar tasks with fewer data and less computational resources. Instead of training a CNN from scratch, we use a model like VGG16, ResNet, or MobileNet as a feature extractor or fine-tune it for our specific classification task.

**Objective:**

 To build a ConvNet using a pre-trained deep learning model.

 To classify images with improved accuracy and reduced training time.

 To understand and implement feature extraction and fine-tuning in Transfer Learning.

**Steps Overview:**

 Import Required Libraries

 Load and Preprocess the Dataset

 Load a Pre-trained Model (e.g., VGG16/ResNet50)

 Freeze Base Layers or Set Trainable as Needed

 Add Custom Classification Head

 Compile and Train the Model

 Evaluate Model Performance

 Save and Load the Trained Model

**Dataset:**

[*https://storage.googleapis.com/mledu-datasets/cats\_and\_dogs\_filtered.zip*](%20https:/storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip)

**Implementation:**

*# Step 1: Import Libraries*

*import tensorflow as tf*

*from tensorflow.keras.applications import VGG16*

*from tensorflow.keras.models import Model*

*from tensorflow.keras.layers import Dense, Flatten, Dropout*

*from tensorflow.keras.preprocessing.image import ImageDataGenerator*

*import matplotlib.pyplot as plt*

*import os*

*import zipfile*

*# Step 2: Download and Extract Dataset*

*\_URL = 'https://storage.googleapis.com/mledu-datasets/cats\_and\_dogs\_filtered.zip'*

*# Download the zip file*

*path\_to\_zip = tf.keras.utils.get\_file('cats\_and\_dogs\_filtered.zip', origin=\_URL, extract=False)*

*# Extract it manually*

*with zipfile.ZipFile(path\_to\_zip, 'r') as zip\_ref:*

*zip\_ref.extractall('/tmp')*

*# Define correct paths for the train and validation directories*

*PATH = '/tmp/cats\_and\_dogs\_filtered'*

*train\_dir = os.path.join(PATH, 'train')*

*val\_dir = os.path.join(PATH, 'validation')*

*# Step 3: Preprocess Data using ImageDataGenerator*

*IMAGE\_SIZE = (224, 224)*

*BATCH\_SIZE = 32*

*train\_gen = ImageDataGenerator(rescale=1./255)*

*val\_gen = ImageDataGenerator(rescale=1./255)*

*train\_data = train\_gen.flow\_from\_directory(train\_dir, target\_size=IMAGE\_SIZE, batch\_size=BATCH\_SIZE, class\_mode='binary')*

*val\_data = val\_gen.flow\_from\_directory(val\_dir, target\_size=IMAGE\_SIZE, batch\_size=BATCH\_SIZE, class\_mode='binary')*

*# Step 4: Load Pre-trained Model (VGG16)*

*base\_model = VGG16(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3))*

*# Freeze all layers of the base model*

*for layer in base\_model.layers:*

*layer.trainable = False*

*# Step 5: Add Custom Layers on Top*

*x = Flatten()(base\_model.output)*

*x = Dense(512, activation='relu')(x)*

*x = Dropout(0.5)(x)*

*predictions = Dense(1, activation='sigmoid')(x)*

*# Final model*

*model = Model(inputs=base\_model.input, outputs=predictions)*

*# Step 6: Compile Model*

*model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])*

*# Step 7: Train the Model*

*history = model.fit(train\_data, validation\_data=val\_data, epochs=5)*

*# Step 8: Plot Training Accuracy*

*plt.plot(history.history['accuracy'], label='Train Accuracy')*

*plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')*

*plt.title('Model Accuracy Over Epochs')*

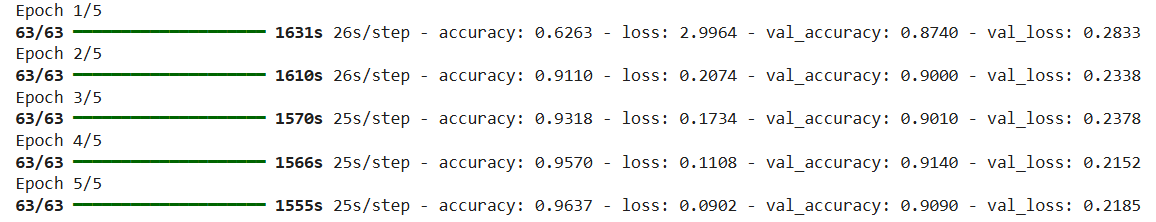
*plt.xlabel('Epochs')*

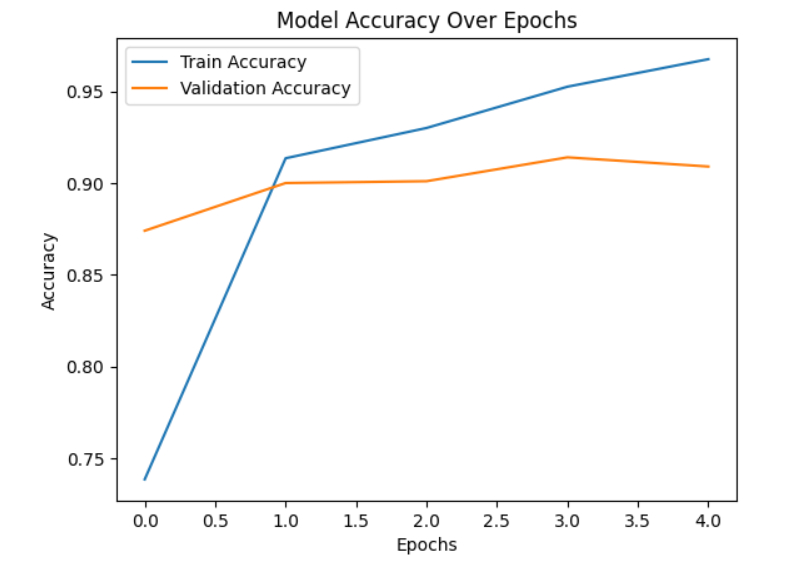
*plt.ylabel('Accuracy')*

*plt.legend()*

*plt.show()*

**Output:**

****

****

**Conclusion:**

Using Transfer Learning significantly reduces training time while achieving high accuracy by leveraging learned features from large datasets. This approach is ideal for image classification tasks, especially when working with limited data. The model can be further fine-tuned by unfreezing some layers of the base model to enhance performance.

In this project, we successfully built an image classifier to distinguish between **cats and dogs** using **transfer learning** with the **VGG16** model. Here's a summary of what we achieved:

**What We Did**

1. **Downloaded and prepared the dataset** from TensorFlow's public source.
2. **Preprocessed the data** using ImageDataGenerator to normalize pixel values.
3. **Loaded a pre-trained VGG16 model** without the top layers to leverage powerful feature extraction.
4. **Added custom dense layers** for binary classification (cats vs dogs).
5. **Trained the model** on the dataset with frozen VGG16 layers to prevent overfitting.
6. **Evaluated the model** using accuracy and plotted the results for better visualization.

**Benefits of Transfer Learning**

* Saved training time by using **pre-trained weights**.
* Improved accuracy by using **features learned from a large dataset (ImageNet)**.
* Reduced need for a large dataset and computational resources.

Transfer learning is a powerful technique in deep learning, especially when:

* You have limited data.
* You want to quickly build high-performance models.
* You want to use proven architectures (like VGG16, ResNet, etc.).